Leveraging Prior Knowledge for Protein-Protein Interaction Extraction with Memory Network

Huiwei Zhou, Zhuang Liu, Shixian Ning, Yunlong Yang
School of Computer Science and Technology, Dalian University of Technology, Dalian, People’s Republic of China

Abstract—Extracting Protein-Protein Interaction (PPI) from biomedical literature is an essential step towards precision medicine. This paper proposes a novel PPI extraction approach, which leverages prior knowledge about protein-protein pair with memory network. The proposed memory network-based model (MNM) captures important context clues related to knowledge representations learned from Knowledge Bases (KBs). We perform 5-fold cross validation on the BioCreative VI PPI training set and achieve a F1-score of 59.67%, which demonstrates the effectiveness of memory network-based approach for the PPI extraction.

Keywords—protein-protein interaction; relation extraction; memory network; prior knowledge

I. INTRODUCTION

With the rapidly growing of biomedical literature, it is becoming urgent and significant for natural language processing (NLP) experts to develop entity relation automatic extraction technique. The BioCreative VI proposes a challenge task of automated PPI extraction from the biomedical literature, which aims to extract interaction relations between protein entities mentioned within a document. Traditional feature-based relation classification methods [1-4] apply machine learning technique to learn models with the one-hot represented lexical and syntactic features. However, these methods need extensive feature engineering and complicate linguistic analysis, which suffer from time consuming and may lead to error propagation. Recently, deep learning techniques have achieved great success in relation extraction tasks. Without the effort of feature engineering, deep neural networks could effectively extract semantic information for relation extraction. Zeng et al. [5] first employ Convolutional Neural Networks (CNN) [6] to capture the lexical and position information for relation classification, and get better performance than the traditional feature-based classifiers. Peng et al. [7] apply CNN for PPI extraction. Except the embeddings vector of each word in the sentence, they also add the embeddings of the head of each word. Gu et al. [8] adopt CNN to integrate context information and dependency information for chemical-disease relation (CDR) extraction.


All methods mentioned above use texts as resources. Nevertheless, biomedical experts have built many large-scale KBs, which contains PPI triples of protein entity pairs and their interaction relation, such as IntAct [14], BioGrid [15], UniProt [16]. These prior knowledge are crucial resources for PPI extraction.

Recently, neural-based representation learning (RL) methods are also applied to encode relational knowledge with low-dimensional embeddings of entities and relations. Many methods have been proposed for knowledge representations (KR) learning, among which translation-based models [17-19] are simple and effective with the state-of-the-art performance. TransE [17] is a typical translation-based method, which regards a relation r as a translation from the head entity h to the tail entity t with the h + r = t in the embedding space, if the triple (h, r, t) holds.

This paper employs TransE to learn embeddings of protein entities and relations from KBs, which are then introduced to a memory network [20-21] to combined with context embeddings. Experiments on the BioCreative VI PPI dataset show that our approach could be effectively leverage prior knowledge to improve PPI extraction performance. Our approach does not rely on feature engineering and is computationally efficient.

II. METHODS

A. Preprocessing

Each protein pair in a given document is regarded as a candidate instance except that the two proteins occur over more than one sentence. Our model takes word embeddings of sequences between protein pairs and knowledge representations learned from KBs as input to calculate the probability of interaction relation. The numbers in the context are replaced by a specific string, such as “NUMBER”. And some special characters, such as “**”, are removed. Additionally, we replace all the entity mentions as “gene0” except the pair entities need to classify. Finally, based on the positive statistics of training data, we select the protein pairs, whose distance is more than 5 and less than 50 characters, to be
the final instances. We use Word2Vec tool\(^1\) [22] to pre-train word embeddings on the datasets (about 9,308MB) downloaded from PubMed2.

B. Knowledge representation learning

We employ TransE model\(^2\) [17] to learn knowledge representations based on the entity-relation triples in protein KBs IntAct [14], BioGrid [15] and UniProt [16].

Note that entity embeddings are initialized with the averaged embeddings of words contained in entity mention and relation embeddings are initialized randomly. We introduce such prior knowledge, embeddings of entity and relation, into the memory network to improve the extraction performance.

C. Architecture of the approach

Memory network is a recurrent attention model over a possibly large external memory which allows to read and write to [20]. To leverage entity embeddings of protein pair, two memory networks are applied to pay attention to the two entity embeddings respectively to select important context words as shown in Fig 1. The two memory networks share the same set of parameters to learn the weights of the context words between the two entities. In addition, the attention mechanism in the two memory networks also share the same parameters, which could enable the two entities to communicate with each other, just like using same eyes to watch two things that may be relevant.

In each memory network, there are two computational layers, each of which contains an attention layer and a linear layer. The outputs of the two networks are concatenated with relation embeddings learned from KBs for relation classification.

D. Memory network model (MNM)

a) Attention mechanism: Given a word sequence between a pair of protein entities in a document \(s = \{w_1, w_2, ..., w_j, ..., w_{n}\}\), the corresponding word embeddings \(\{e_1, e_2, ..., e_i, ..., e_n\}\) is regard as the memory \(m \in \mathbb{R}^{d \times n}\), where \(e_i \in \mathbb{R}^d\) is a \(d\)-dimensional word vector. As for each attention layer, one entity embedding is concatenated to each piece of memory \(m_i\) to select important evidences as follows:

\[
g_i = \tanh(W_s[m_i; e_{entity}] + b_s)\]

where \([m_i; e_{entity}]\) denotes the concatenation of \(m_i \in \mathbb{R}^{d \times 1}\) and protein embedding \(e_{entity} \in \mathbb{R}^{d \times 1}\), \(W_s \in \mathbb{R}^{1 \times 2d}\), and \(b_s \in \mathbb{R}^{1 \times 1}\).

After obtaining \([g_1, g_2, ..., g_n]\), the attention weight of each word can be defined as follows:

\[
\alpha_i = \frac{\exp(g_i)}{\sum_{j=1}^{n} \exp(g_j)}
\]

The attention layer output \(output_a \in \mathbb{R}^{d \times 1}\) is then calculated as a weighted sum of each piece of memory in \(m\):

\[
output_a = \sum_{i=1}^{n} \alpha_i m_i
\]

b) Dimension-wise max pooling: To take an optimal over the context embeddings and entity embedding, we do a dimension-wise max pooling on the attention layer output \(output_a\) and the linear transformation of entity embedding \(e_{entity}\). And the max pooling vector is considered as the new entity embedding \(e_{entity}\) for the next layer. For the other entity, the same operations are applied. Finally, we concatenate the two max pooling vectors of the second computational layer and the relation embeddings learned from KBs, and fed it to a softmax layer to perform relation classification.

c) Location impaction: Following Sukhbaatar [20], we control the input percentage of each piece of memory by its relative distance to the entity mention. Each percentage is calculated as follows:

\[
per_{i,k} = (1 - p_j / n) - (k / d)(1 - 2p_j / n)
\]

---

1 https://code.google.com/p/word2vec/
2 http://www.ncbi.nlm.nih.gov/pubmed/
Where \( n \) is sequence length, \( k \) is the number of current layer, \( p_i \) is the relative distance between current word and entity mention, \( d \) is the dimension of word embeddings. Therefore, the actual memory vector is computed with \( m_{i,k} = e_i \odot \per_{i,k} \).

### III. RESULTS AND DISCUSSION

We use the BioCreative VI PPI training dataset which has annotated some protein pairs that have relation, and perform a 5-fold cross validation on it. Protein entity recognition and normalization are accomplished with GNormPlus [23] toolkits. The dimension of word, entity, and relation embedding are all 100. The memory network model is trained by using Adam technique [24] with a learning rate 0.001. The whole framework is developed by PyTorch.

#### A. Effects of prior knowledge

In the experiments, the proposed MNM is compared with the following baseline methods:

**AE (Averaged Entity Embeddings):** This method represents entities as an average of their constituting word embeddings, but does not concatenate relation embeddings to the outputs of memory networks. Only entity embeddings are employed. And relation embeddings are not used.

**TE (TransE-based Entity Embeddings):** This method employs TransE-based entity embeddings learned from KBs, but does not concatenate relation embeddings to the outputs of memory networks. Only entity embeddings are employed. And relation embeddings are not used.

**AE-TR (Averaged Entity Embeddings and TransE-based Relation Embeddings):** This method represents entities as an average of their constituting word embeddings, and concatenate TransE-based relation embeddings learned from KBs to the outputs of memory networks. Both entity embeddings and relation embeddings are applied.

Table I lists the comparison results. Seen from the table, the proposed MNM outperforms all three baseline methods. Among the three baselines, the best one is AE-TR, which employs relation embeddings learned from KBs. Actually, AE-TR is similar to MNM except that entity embeddings are not learned from KBs.

Compared with AE, AE-TR simply add the relation embeddings and the \( F_1 \)-score is improved by 12.71%, which indicates that relation embeddings could provide effective clues about PPI relations to classifier. In MNM, both entity embeddings and relation embeddings are learned from KBs, and \( F_1 \)-score is 59.67%, which is 1.04% higher than AE-TR. That illustrates structured information contained in TransE-based entity embeddings is more effective than the implicit semantic information expressed by word embeddings for relation classification. However, TE makes \( F_1 \)-score drop 1.46% compared with AE. It perhaps because that without relation embeddings, structured prior knowledge in TransE-based entity embeddings is hard been captured by our model.

Note that if a protein entity is absent in KBs, the entity embedding is randomly initialized in MNM training. And for protein pairs not found in KBs, relation embedding is initialized as zero vectors.

#### TABLE I. EFFECTS OF PRIOR KNOWLEDGE

<table>
<thead>
<tr>
<th>Prior knowledge</th>
<th>( F_1 )-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>AE</td>
<td>45.92</td>
</tr>
<tr>
<td>TE</td>
<td>44.46</td>
</tr>
<tr>
<td>AE-TR</td>
<td>58.63</td>
</tr>
<tr>
<td>MNM</td>
<td>59.67</td>
</tr>
</tbody>
</table>

#### B. Sharing attention operation

Table II shows the effect of sharing attention operation. As can be seen from the table, not sharing attention operation causes \( F_1 \)-score to drop 0.66%. We believe that different attention operations to the two relevant entities would introduce more noises, which are not helpful to relation classification.

#### TABLE II. SHARING ATTENTION OPERATION

<table>
<thead>
<tr>
<th>Sharing or not</th>
<th>( F_1 )-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not sharing</td>
<td>59.01</td>
</tr>
<tr>
<td>Sharing</td>
<td>59.67</td>
</tr>
</tbody>
</table>

#### C. Effects of computational layer number

We try several computational layers and the results are shown in Table III. Our MNM with different numbers of computational layers are expressed as MNM (k), where \( k \) is the number of computational layers. Among all models from single computational layer to four, we can observe that two and three layers get better performance. The best performance is achieved when the model contains two computational layers.

#### TABLE III. EFFECTS OF COMPUTATIONAL LAYER NUMBER

<table>
<thead>
<tr>
<th>Computational layer number</th>
<th>( F_1 )-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNM(1)</td>
<td>59.07</td>
</tr>
<tr>
<td>MNM(2)</td>
<td>59.67</td>
</tr>
<tr>
<td>MNM(3)</td>
<td>59.49</td>
</tr>
<tr>
<td>MNM(4)</td>
<td>58.83</td>
</tr>
</tbody>
</table>

#### D. Comparison with LSTM

Table IV compares our memory network-based model (MNM) with bidirectional long short-term memory (Bi-LSTM) networks [13], which is the other state-of-art networks in relation classification. Different from [13], we introduce knowledge representations learned from KBs to our Bi-LSTM model. Each entity embeddings is concatenated with context word embeddings \( \{e_1, e_2, ..., e_n\} \) as each direction input.

Attention mechanism is applied to produce a weight vector, and merge hidden representations from each time step into a sequence-level vector. The concatenation of bidirectional sequence-level vectors is concatenated with relation
embeddings further for relation classification. From the table we can see that the proposed memory network-based model (MNM) is more powerful in taking advantage of prior knowledge than LSTM architecture.

**TABLE IV. COMPARISON WITH LSTM**

<table>
<thead>
<tr>
<th>Model structure</th>
<th>Evaluation Measurement (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>p</td>
</tr>
<tr>
<td>Bi-LSTM</td>
<td>49.77</td>
</tr>
<tr>
<td>MNM</td>
<td>50.91</td>
</tr>
</tbody>
</table>

E. Attention visualization

To better demonstrate the effectiveness of attention mechanism, attention weights of two example sequences are visualized in the form of heat map in Fig. 2. For each sequence, the upper visible layers shows the weights of the context words toward Entity1, and the lower shows the weights of the context words toward Entity2. In the first example, “found” and “phosphorylated” have the maximum weights when paying attention to the entity “28984” and entity “207” respectively. As for the second example, “controls” has the maximum weight no matter which entity we pay attention to. From Fig. 2, we observe that our MNM could identify the important word effectively.

![Fig. 2. Visualization of attention weight by heat map.](image)

IV. CONCLUSION

In this paper, we propose a memory network-based PPI extraction approach. The proposed approach calculates the weight of context words toward two protein entities by leveraging prior knowledge with attention mechanism in two memory networks. The experimental results show that the proposed memory network-based approach is better than LSTM architecture, and knowledge representations learned from KBs could significantly improve the PPI performance.

ACKNOWLEDGMENT

This research is supported by National Natural Science Foundation of China (Grant No. 61772109) and The Ministry of education of Humanities and Social Science research and planning Fund of China (Grant No. 17YJA740076).

REFERENCES